Chapter 10. Advanced DOS Commands

Contents

Notes:

Introduction

This chapter explains how to use the advanced DOS commands. You can use advanced DOS commands to:

- Set options for the Asynchronous Communications Adapter.
- Define a remote device as your primary console.
- Sort text data.
- Search files for occurrences of specified strings of text.
- Display a screen full of data at a time.
- Set new system prompt.
- Set the system environment.
- Convert .EXE files to .COM files.

Redirection of Standard Input and Output Devices

The DOS standard input and output device redirection feature allows a program to receive its input from a source other than the keyboard (standard input), or direct its output to a device other than the display screen (standard output).

The standard input and output devices can be redirected to or from files or other devices by the following DOS command line parameters:

>[*d*:][*path*]*filename*

Causes *filename* to be created (or truncated to zero length) and then assigns standard output to that file. All output that would normally have gone to the screen from the command is placed in the file.

>>[*d*:][*path*]*filename*

Causes *filename* to be opened (created if necessary) and positions the write pointer at the end of the file so that all output is appended to the file.

<[*d*:][*path*]*filename*

Causes standard input to be assigned to *filename*. All input to the program comes from this file instead of from the keyboard.

CAUTION

When using this method of providing input to a program, be sure *all* of the program's input is in the file. If the program attempts to obtain more input after end-of-file is reached, DOS is unable to supply the input, and processing will stop. You can return to the DOS prompt by entering Ctrl-Break.

Note: If an application does not use DOS function calls to perform standard input and/or output (for example you put text directly into the video buffer), then redirection will not work for that application.

Example: In this example, the output of the DIR command is sent to the printer:

```
DIR >PRN
```

In this example, the output of the DIR command is sent to file DIRLIST:

```
DIR >DIRLIST
```

In the following example, program MYPROG will receive its input from file INPUT.TXT, instead of from the keyboard:

```
MYPROG <INPUT.TXT
```

Piping of Standard Input and Output

The DOS piping feature allows the screen output of one program to be used as the keyboard input to another program. DOS uses temporary files to hold the input and output data being piped. These temporary files are created in the root directory of the default drive and have the form:

```
%PIPEx.$$$
```

The programs being piped must use care not to cause the piping files to be erased or modified.

Piping is the chaining of programs with automatic redirection of standard input and output (refer to "Redirection of Standard Input and Output Devices" in this chapter for additional information). The names of the programs to be chained are separated by the vertical bar (¦) character on the command line.

The following are typical examples of using the piping feature for a program that does all of its input and output to the standard devices (screen and keyboard). For example, if the program named SORT read all of it's standard input, sorted it, and then wrote it to the standard output device, the command:

```
DIR¦SORT
```

Would generate a sorted directory listing. The causes all standard output generated by the DIR command to be sent to the standard input of the SORT program.

To send the sorted directory to a file, you would type:

```
DIR¦SORT>FILE
```

If you wish the file to contain only the directory entries for sub-directories, you could enter:

```
DIR¦FIND  "DIR" ¦ SORT>FILE
```

DOS Filters

A filter is a program or command that reads data from a standard input device, modifies the data, then writes the result to a standard output device. Thus, the data has been "filtered" by the program. For example, one of the filters on your DOS diskette is called SORT. SORT reads input from the standard input device (normally the keyboard), sorts the lines of data, then writes the sorted results to the standard output device (normally the screen). With the redirection capabilities described earlier in this chapter, you can cause SORT to receive its input from some other source, and to send its output to a different destination. For example,

```
SORT <MYFILE >RESULT
```

Will cause SORT to read the file MYFILE, sort the lines within it, and write the sorted output to file RESULT.

By using the piping feature, you can cause a filter to receive its input from the output of another command, or to send its output to the input of another command. For example,

```
DIR | SORT
```

Causes the output listing from the DIR command to be used by SORT as its input. The listing will be sorted and the result displayed on the screen.

There are three filters on your DOS diskette, and they are described as individual commands in this chapter. They are:

SORT	Sorts text data.
FIND	Searches files for occurrences of specified strings of text.
MORE	Displays a screen full of data at a time, then pauses with the message —MORE—.

You can easily add your own filter to the filters that have been supplied; just write a program that reads its input from the standard input device, and writes its output to the standard output device.

> **Note:** If an application does not use DOS function calls to perform standard input and/or output (for example you put text directly into the video buffer), then filters will not work for that application.

Detailed Descriptions of Advanced DOS Commands

This section presents a detailed description of how to use the advanced DOS commands. The commands appear in alphabetical order; each with its purpose, format, and type. Examples are provided where appropriate. For "Information Common to All DOS Commands" refer to Chapter 6.

Invoking a Secondary Command Processor

If you wish to invoke a secondary command processor, the following syntax should be used:

```
COMMAND [d:][path] [/P] [/C string]
```

Where d:path will be the directory searched for the command processor to be loaded, /P causes the new copy to become permanent in memory, and /C string allows you to pass a command line (string) as a parameter. The command line will be interpreted and acted upon as if you had entered it as a normal command. For example, COMMAND /C DIR B: causes a secondary command processor to be loaded, and it executes the command DIR B:.

When a secondary command processor has been loaded, you can cause it to return to the previous level of command processor by issuing the special command EXIT. Note that if you used the /P parameter, it will not return to the previous level (refer to Appendix F for additional information).

CTTY (Change Console) Command

Purpose: Changes the standard input and output console to an auxiliary console, or restores the keyboard and screen as the standard input and output devices.

Format: CTTY *device-name*

Type: Internal *** External

Remarks: Defines the device to be used as the primary console. Specifying AUX, COM1, or COM2 causes DOS to use that device as the primary console. Specifying CON resets the standard input and output device to the primary console.

Example: In this example, the command causes DOS to use the AUX device for its screen and keyboard operations:

```
CTTY AUX
```

In this example, the command reverses the previous assignment, causing DOS to switch back to the standard screen and keyboard for its operations:

```
CTTY CON
```

Note: The CTTY command accepts the name of *any* character-oriented device to allow you to install your own device drivers, and to specify their device names. You must be certain that the named device is capable of both input and output operations. For example, you should not specify the name of a printer, because DOS will attempt to read from that device.

EXE2BIN Command

Purpose: Converts .EXE files that have no segment fixup to a form that is compatible with .COM programs. This results in a saving of diskette space and faster program loading.

Format: EXE2BIN [*d*:][*path*]*filename*[.*ext*]]

[*d*:][*path*][*filename*[.*ext*]]

Type:

Internal	External
	***

Remarks: The file named by *filespec* is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file, [*d*:]*filename*[.*ext*]. If you do not specify a drive, the drive of the input file is used. If you do not specify an output filename, the input filename is used. If you do not specify a filename extension in the output filename, the new file is given an extension of .BIN. If you do not specify a path, the current directory is used.

The input must be in valid .EXE format as produced by the linker. The *resident*, or actual code and data, part of the file must be less then 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on the specified initial CS:IP:

- If CS:IP is not specified in the program (the .EXE file contains 0:0), a pure binary conversion is assumed. If segment fixups are necessary (the program contains instructions requiring segment relocation), you are prompted for the fixup value. This value is the absolute segment at which the program is to be loaded.

  In this case, the resultant program is usable only when loaded at the absolute memory address specified by a user application. The DOS command processor will not be capable of properly loading the program.

- If CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a COM file, with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as COM files must be segment relocatable; that is, they must assume the entry conditions explained in Appendixes B-G. In this case, once the conversion is complete, you may rename the resultant file to a .COM extension. Then, the command processor is capable of loading and executing the program in the same manner as the .COM programs supplied on your DOS diskette.

If CS:IP does not meet one of these criteria, or if it meets the COM file criterion but has segment fixups, the following message is displayed:

```
File cannot be converted
```

This message is also displayed if the file is not a valid .EXE file.

To produce standard COM files with the assembler, you must both use the assembler statement ORG to set the location pointer of the file at 100H and specify the first location as the start address. (This is done in the END statement.) Also, the program must not use references that are defined only in other programs. For example, with the IBM Personal Computer MACRO Assembler:

```
ORG 100H
START:
•
•
•
END START
```

EXE2BIN resides on your DOS Supplemental Program diskette.

FIND Filter Command

Purpose: This filter sends to the standard output device all lines from the filenames specified in the command line that contain the specified string.

Format: FIND [/V][/C][/N]*string*[[*d:*][*path*]*filename*[.*ext*]...]

Type:

Internal	External
	***

Remarks: The **/V** parameter causes all lines *not* containing the *string* to be displayed.

The **/C** parameter causes FIND to display only a count of the number of matching occurrences of *string* in each file, without displaying the matching lines from the file.

The **/N** parameter causes the relative line number of each matching line to be displayed ahead of the line from the file.

The string should be enclosed in double quotes. Two quotes in succession are taken as a single quote.

Global filename characters are not allowed in the filenames or extensions.

FIND Filter Command

Examples:

```
A>FIND "Fool's Paradise" book1.txt book2.txt book3
```

Will output all lines from the book1.txt, book2.txt, and book3 (in that order) that contain the string "Fool's Paradise". Or,

```
A>DIR B: ¦ FIND /V "DAT"
```

Will output the names of all the files in drive B that do not contain the string DAT.

MORE Filter Command

Purpose: This filter reads data from the standard input device, and sends one screen-full of data to the standard output device, and then pauses with the message **——More——**.

Format: MORE

Type:

Internal	External
	***

Remarks: Pressing any character key causes another screen-full of data to be written to the standard output device. This process continues until all input data is read.

Example: In this example, the command line will display the contents of file TEST.ASM one screen-full at a time. When the screen is full, the message **—More—** appears on the bottom line. You can press any key to see the next screen-full:

```
MORE <TEST.ASM
```

PROMPT (Set System Prompt) Command

Purpose: Sets a new system prompt.

Format: PROMPT [*prompt-text*]

Type:

Internal	External
	***

Remarks: All text on the PROMPT command line is taken by DOS to be the new system prompt. If no parameter is specified, the normal DOS prompt is assumed. Special meta-strings can be imbedded in the text in the form **$***c*.

Where *c* is one of the following:

$	The "$" character.
t	The time.
d	The date.
p	The current directory of the default drive.
v	The version number.
n	The default drive.
g	The ">" character.
l	The "<" character.
b	The "|" character.
q	the "=" character.
h	A backspace and erasure of the previous character.
e	The ESCape character.
–	The CR LF sequence (go to beginning of new line on the display screen).

PROMPT (Set System Prompt) Command

Any other character is treated as a null character—no action is taken on it by the PROMPT command.

Example: In this example, the command would set the normal DOS prompt:

```
PROMPT $n$g
```

In this example, the command would set ABC as the system prompt:

```
PROMPT ABC
```

In this example, the command would set a two line prompt that displays:

```
Time = (current time)

Date = (current date)

PROMPT Time = $t$ ____ Date=$d
```

If you wish to create a prompt that begins with any of the DOS command delimiters (such as semicolon, blank, etc.), you can precede that character with a null meta-string. In this case, the character will be treated as the first character of the prompt, rather than as a delimiter between the word PROMPT and its parameter. For example:

```
PROMPT $A;ABC
```

PROMPT (Set System Prompt) Command

Causes the PROMPT command to interpret the $A as a null character, because A is not one of the defined characters in the above list. All characters following the null character will become the new system prompt.

SET (Set Environment) Command

Purpose: This command inserts strings into the command processor's environment. The entire series of strings in the environment is made available to all commands and applications.

Format: Set [*name*=[*parameter*]]

Type: Internal ***   External

Remarks: The entire string (beginning with *name*) is inserted into a block of memory reserved for environment strings. Any lowercase letters in the name are converted to uppercase letters when added to the environment; the remainder of the line is inserted as you entered it. If the name already existed in the environment, it is replaced with the new *parameter*.

If the SET command is entered with no *name* specified, then the current set of environment strings will be displayed.

If a *name* is specified, but the *parameter* is not specified, then the current occurrence of *name*=*parameter* is removed from the environment.

SET (Set Environment) Command

The environment (series of names and parameters) is made available to all DOS commands and applications (see the "Program Segment Prefix description" in Appendix E). You can display the current environment contents by entering a SET command with no parameters. You can select the strings in the environment. For example, entering:

```
SET abc=xyz
```

Will add the string ABC=xyz to the other strings already in the environment (note the conversion of abc to uppercase ABC). In this way, it is possible for you to enter keywords and parameters that are not meaningful to DOS, but can be found and interpreted by applications that are designed to examine the environment.

Example: This example will add the string PGMS=\LEVEL2 to the environment. When an application program receives control, it could search the environment for the name PGMS, and use the supplied parameter as the directory name to use for its files:

```
SET PGMS=\LEVEL2
```

The following example would remove PGMS=\LEVEL2 from the environment:

```
SET PGMS=
```

Notes:

1. DOS automatically adds any PROMPT or PATH commands to the environment when you enter them. You do not need to use the SET command to add either of these two commands to the environment.

2. The first string in the environment (placed there by DOS when it starts up) will always be a COMSPEC = parameter. That parameter describes the path that DOS uses to reload the command processor when necessary.

3. If you have *not* loaded a program that remains resident (such as MODE, PRINT, GRAPHICS, etc.), DOS expands the environment string area to hold additional strings. If you *have* loaded a program that remains resident, DOS is unable to expand the environment area beyond 127 bytes or if environment area has already expanded beyond 127 bytes when you load a program that is to remain resident, DOS is unable to expand the environment area beyond that point. The message **Out of environment space** appears if you issue a SET command that would cause the combined environment strings to exceed 127 bytes.

SORT Filter Command

Purpose: This filter command reads data from the standard input device, sorts the data, then writes the data to the standard output device.

Format: SORT [/R] [/+ *n*]

Type:	Internal	External
		***

Remarks: Sorts are done using the ASCII collating sequence. Tab characters are not expanded with blanks.

The **/R** parameter will reverse the sort, for example make "Z" come before "A."

The /+ *n* parameter is an integer that starts the sort with column *n*. If no parameters are specified, the sort starts with column 1. The maximum file size that can be sorted is 63K.

SORT Filter Command

Example: In this example, the command line will read the file UNSORT.TXT, do a reverse sort, then write the output to file SORT.TXT:

```
A>SORT /R <UNSORT.TXT >SORT.TXT
```

In the next example, the command line causes the output of the directory command to be piped to the SORT filter. The SORT filter will sort starting with column 14 (this is the column the file size starts in), then send the output to the console. Thus, a directory sorted by file size will be the result:

```
A>DIR ¦ SORT /+14
```

Summary of Advanced DOS Commands

The following chart is provided for quick reference. The section called "Format Notation" at the beginning of Chapter 6 explains the notation used in the format of the commands.

> **Note:** In the column labeled **Type**, the I stands for Internal and the E stands for External.

Command	Type	Purpose	Format
CTTY	I	Change to an auxiliary console	CTTY *device-name*
EXE2BIN	E	Converts .EXE files to .COM format	EXE2BIN [*d*:][*path*]*filename*[.*ext*]] [*d*:][*path*][*filename*[.*ext*]]
FIND	E	Searches files for strings of text	FIND [/V][/C][/N]*string*[*d*:] [*path*]*filename*[.*ext*]...]
MORE	E	Displays a screen full of data	MORE
PROMPT	E	Set new prompt	PROMPT [*prompt-text*]
SET	I	Inserts strings into the command processor's environment	SET [*name*=[*parameter*]]
SORT	E	Sorts text data	SORT [/R] [/+*n*]

Figure 4. DOS Advanced Commands